DOHRNII

Dohrnii Token Contract Review

Security Assessment Report

Version: 2.0

August, 2026

Contents

Introduction

Sigma Prime was commercially engaged to perform a time-boxed security review of the Dohrnii components in scope. The review focused solely on the security aspects of the Solidity implementation of the contract, though general recommendations and informational comments are also provided.

Disclaimer

Sigma Prime makes all effort but holds no responsibility for the findings of this security review. Sigma Prime does not provide any guarantees relating to the function of the components in scope. Sigma Prime makes no judgements on, or provides any security review, regarding the underlying business model or the individuals involved in the project.

Document Structure

The first section provides an overview of the functionality of the Dohrnii components contained within the scope of the security review. A summary followed by a detailed review of the discovered vulnerabilities is then given which assigns each vulnerability a severity rating (see Vulnerability Severity Classification), an *open/closed/resolved* status and a recommendation. Additionally, findings which do not have direct security implications (but are potentially of interest) are marked as *informational*.

The appendix provides additional documentation, including the severity matrix used to classify vulnerabilities within the Dohrnii components in scope.

Overview

Dohrnii (DHN) is a fixed-supply ERC-20 token deployed behind a UUPS proxy with a blacklist mechanism managed through separate management roles. A blacklisted address can neither send nor receive DHN.

The review has focused on a correct implementation of the UUPS proxy standard, the correct mechanics of the blacklist feature, and the correct and expected functionality of the role management restrictions and their expected trust boundaries.

Security Assessment Summary

Scope

The review was conducted on the files hosted on the dohrnii-dev/Dohrnii_token_project repository.

The scope of this time-boxed review was strictly limited to the following file at commit cd26674:

- `contracts/DohrniiToken.sol`

The deployment and upgrade scripts, the test suite and the operator documentation were not part of the scope, though they were read where a finding in the contract depended on the procedure or the guarantee they describe.

The fixes of the identified issues were assessed at commit 8a28d88.

Note: third party libraries and dependencies were excluded from the scope of this assessment.

Approach

The security assessment covered components written in Solidity.

For the Solidity components, the manual review focused on identifying issues associated with the business logic implementation of the contracts. This includes their internal interactions, intended functionality and correct implementation with respect to the underlying functionality of the Ethereum Virtual Machine (for example, verifying correct storage/memory layout).

Additionally, the manual review process focused on identifying vulnerabilities related to known Solidity anti-patterns and attack vectors, such as re-entrancy, front-running, integer overflow/underflow and correct visibility specifiers.

To support the Solidity components of the review, the testing team may use the following automated testing tools:

- Aderyn: `https://github.com/Cyfrin/aderyn`
- Slither: `https://github.com/trailofbits/slither`
- Mythril: `https://github.com/ConsenSys/mythril`

Output for these automated tools is available upon request.

Coverage Limitations

Due to the time-boxed nature of this review, all documented vulnerabilities reflect best effort within the allotted, limited engagement time. As such, Sigma Prime recommends to further investigate areas of the code, and any related functionality, where majority of critical and high risk vulnerabilities were identified.

The contract in scope is small, delegates the bulk of its behaviour to audited OpenZeppelin implementations and adds no arithmetic of its own, and the testing team identified no critical or high risk vulnerabilities within it. The residual risk sits almost entirely in the privileged surface rather than in the token logic: the reach of each role, the process by which

each role is transferred, and the accuracy of the operator documentation that describes both. Sigma Prime recommends that any further work concentrate on that surface, and in particular on the key-management and operational procedures that surround the owner wallet, which are outside the scope of this review but on which the safety of the deployment depends.

Findings Summary

The testing team identified a total of 4 issues during this assessment. Categorised by their severity:

- Low: 1 issue.
- Informational: 3 issues.

Detailed Findings

This section provides a detailed description of the vulnerabilities identified within the Dohrnii components in scope. Each vulnerability has a severity classification which is determined from the likelihood and impact of each finding by the matrix given in the Appendix: Vulnerability Severity Classification.

A number of additional properties of the components, including optimisations, are also described in this section and are labelled as “informational”.

Each vulnerability is also assigned a **status**:

- ***Open:*** the finding has not been addressed by the project team.
- ***Resolved:*** the finding was acknowledged by the project team and updates to the affected components have been made to mitigate the related risk.
- ***Closed:*** the project team has reviewed and acknowledged the finding, but the recommended remediation has not been implemented. The project team has typically provided a documented rationale supporting this decision, including the approach taken and any associated risk considerations.

Summary of Findings

ID	Description	Severity	Status
DHN-01	Blacklisted Spenders Can Still Move Approved Tokens	Low	Resolved
DHN-02	A Pending Ownership Transfer Never Expires	Informational	Resolved
DHN-03	`UPGRADER_ROLE` Is Granted Without The Protections That Guard Ownership	Informational	Resolved
DHN-04	Miscellaneous General Comments	Informational	Resolved

DHN-01	Blacklisted Spenders Can Still Move Approved Tokens		
Assets	`contracts/DohrniiToken.sol`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Medium	Likelihood: Low

Description

The blacklist restricts a transfer's two balance endpoints but not its caller, so a blacklisted account can still exercise any allowance it holds. Sanctioning an address does not remove its ability to move the balance of an account that has approved it.

The consequence is most severe where the approved spender is a protocol contract that holds allowances but no balance of its own, such as a router. If such a contract is compromised, blacklisting it does not stop the drain, and the remaining alternatives may be impracticable. Blacklisting the `to` address requires identifying it before the transaction lands, which a private mempool prevents, and a fresh address per victim can make the blacklisting impossible due to gas limits. For the same block gas limit reason, blacklisting the `from` addresses can be impossible if the user base is large enough, which is realistic for DeFi protocols. Furthermore, freezing `from` is blacklisting victims rather than the attacker.

`_update()` inspects `from` and `to` only. In a `transferFrom()` call the spender is neither, so a blacklisted spender holding a non-zero allowance may continue to move tokens out of an approving account to any address that is not itself blacklisted.

`contracts/DohrniiToken.sol::DohrniiToken::_update()`

```
function _update(address from, address to, uint256 value) internal virtual override {
    DohrniiTokenStorage storage $ = _getDohrniiTokenStorage();

    if ($.blacklistEnabled) {
        if ($.blacklist[from]) revert DohrniiBlacklistedAddress(from);
        if ($.blacklist[to]) revert DohrniiBlacklistedAddress(to);
        // @audit the caller is never checked, so a blacklisted spender may still call transferFrom()
    }

    super._update(from, to, value);
}
```

Even if this behaviour is considered a design choice, the testing team deems the scenario above too realistic to ignore, and has therefore not filed this issue as informational.

This issue has a medium impact as the exposure is bounded only by the total amount approved to the sanctioned spender, and `setBlacklistedBatch()` allows some of those funds to be protected.

This issue has a low likelihood as it requires the compromise of an external contract that holds broad allowances.

Recommendations

Extend blacklist enforcement to the spender in allowance based transfers, so that a blacklisted spender cannot exercise an allowance.

Resolution

The development team has resolved this issue in commit `8a28d88` by overriding `_spendAllowance()` to revert when the spender is blacklisted. The check precedes the call to `super`, so it also applies where the allowance is unlimited, and `transferFrom()` is the only allowance consuming path in the contract.

`approve()` remains unrestricted in both directions, so an approving account can still revoke an allowance it granted to an address that is later blacklisted.

DHN-02	A Pending Ownership Transfer Never Expires
Assets	`contracts/DohrniiToken.sol` `docs/RUNBOOK.md`
Status	**Resolved:** See Resolution
Rating	Informational

Description

A scheduled ownership transfer remains claimable indefinitely. Once the delay has elapsed the nominated address may accept at any point in the future, so a nomination that is abandoned rather than cancelled leaves a claim on `DEFAULT_ADMIN_ROLE` that remains live indefinitely, unless it is cancelled or replaced.

`@openzeppelin/contracts-upgradeable/access/extensions/AccessControlDefaultAdminRulesUpgradeable.sol::_acceptDefaultAdminTransfer()`

```
function _acceptDefaultAdminTransfer() internal virtual {
    AccessControlDefaultAdminRulesStorage storage $ = _getAccessControlDefaultAdminRulesStorage();
    (address newAdmin, uint48 schedule) = pendingDefaultAdmin();
    if (!_isScheduleSet(schedule) || !_hasSchedulePassed(schedule)) {
        revert AccessControlEnforcedDefaultAdminDelay(schedule);
    }
    // @audit no upper bound on schedule, so the claim stays live for as long as it is not cancelled
    _revokeRole(DEFAULT_ADMIN_ROLE, defaultAdmin());
    _grantRole(DEFAULT_ADMIN_ROLE, newAdmin);
    // ... snip ...
}
```

This issue is rated as informational because the indefinite claim arises from an operational omission, namely leaving a nomination un-cancelled, rather than from a defect in the contract.

Recommendations

Consider bounding the acceptance window onchain so that a lack of cancellation does not leave an indefinite risk.

`docs/RUNBOOK.md` presents a nomination that is never accepted as benign, stating that "an address that never accepts leaves ownership where it is". This is true if the nominee never acts. However, if the nominee eventually accepts, ownership changes. Record cancellation of an abandoned nomination as a mandatory step in `docs/RUNBOOK.md`, and amend the wording so that a nomination which is never accepted is described as a claim that remains live rather than as a state that poses no risks.

Resolution

The development team has resolved this issue in commit `8a28d88` by overriding `acceptDefaultAdminTransfer()` to reject an acceptance made more than `ADMIN_ACCEPT_WINDOW`, currently 30 days, after the nomination becomes acceptable. `docs/RUNBOOK.md` now records cancelling an abandoned nomination as a mandatory step.

The deadline is enforced in the public entry point rather than in `_acceptDefaultAdminTransfer()`. No other path reaches the internal function in the current implementation, so the bound holds, though a future version that calls it directly would not inherit the check.

DHN-03	`UPGRADER_ROLE` Is Granted Without The Protections That Guard Ownership
Assets	`contracts/DohrniiToken.sol` `README.md`
Status	**Resolved:** See Resolution
Rating	Informational

Description

`UPGRADER_ROLE` can replace the entire implementation and is therefore at least as powerful as `DEFAULT_ADMIN_ROLE`, yet it is granted and moved by a single immediate `grantRole()` call with no delay, no acceptance by the recipient and no cancellation window. The protections the contract applies to ownership do not extend to a role that is able to remove them.

Despite its name, `UPGRADER_ROLE` has effective control of the contract because it can change the implementation at any time, and by design it can be a different address than `DEFAULT_ADMIN_ROLE`.

This issue is rated as informational because every grant of `UPGRADER_ROLE` requires the default admin key. `README.md` identifies honest custody of that key as a design trust assumption.

Recommendations

Consider applying protections similar to the ones that already guard `DEFAULT_ADMIN_ROLE` to `UPGRADER_ROLE`. For example, add a delay on upgrades, and a delayed two-step process for granting and moving the role itself.

Resolution

The development team has resolved this issue in commit `8a28d88` by making upgrades a two-step, time-delayed operation. `scheduleUpgrade()` commits to an implementation and its calldata, execution is possible only after `UPGRADE_DELAY` and within `UPGRADE_WINDOW`, and either `UPGRADER_ROLE` or `DEFAULT_ADMIN_ROLE` may cancel. The override preserves the parent's proxy and authorisation checks.

`DEFAULT_ADMIN_ROLE` is also bound by the delay. A pass-through for that role would allow a defect in a future implementation to be corrected without waiting, at the cost of the present guarantee that no key can change the implementation without a day of notice.

DHN-04	Miscellaneous General Comments

Assets	`contracts/DohrniiToken.sol`
	`README.md`
	`docs/RUNBOOK.md`
	`test/*`

Status	**Resolved:** See Resolution
Rating	Informational

Description

This section details miscellaneous findings discovered by the testing team that do not have direct security implications:

1. **Privileged And Proxy Functions Lack Test Coverage**

   Related Asset(s): `test/DohrniiToken.test.ts, test/Upgrade.test.ts`

   `renounceRole()`, `supportsInterface()`, the `onlyProxy` guard and `proxiableUUID()` do not appear in `test/`. No issue was found in these functions; however, adding tests for them is recommended.

   Consider adding tests for each of the functions listed.

Recommendations

Ensure that the comments are understood and acknowledged, and consider implementing the suggestions above.

Resolution

The development team's responses to the raised issues above are as follows.

1. **DHN-04.1 — Privileged And Proxy Functions Lack Test Coverage**

   Resolved in commit `8a28d88`. Tests were added for `renounceRole()`, `supportsInterface()`, the `onlyProxy` guard and `proxiableUUID()`, alongside coverage of the blacklist, admin window and upgrade delay logic introduced by the other fixes.

Appendix A Vulnerability Severity Classification

This security review classifies vulnerabilities based on their potential impact and likelihood of occurance. The total severity of a vulnerability is derived from these two metrics based on the following matrix.

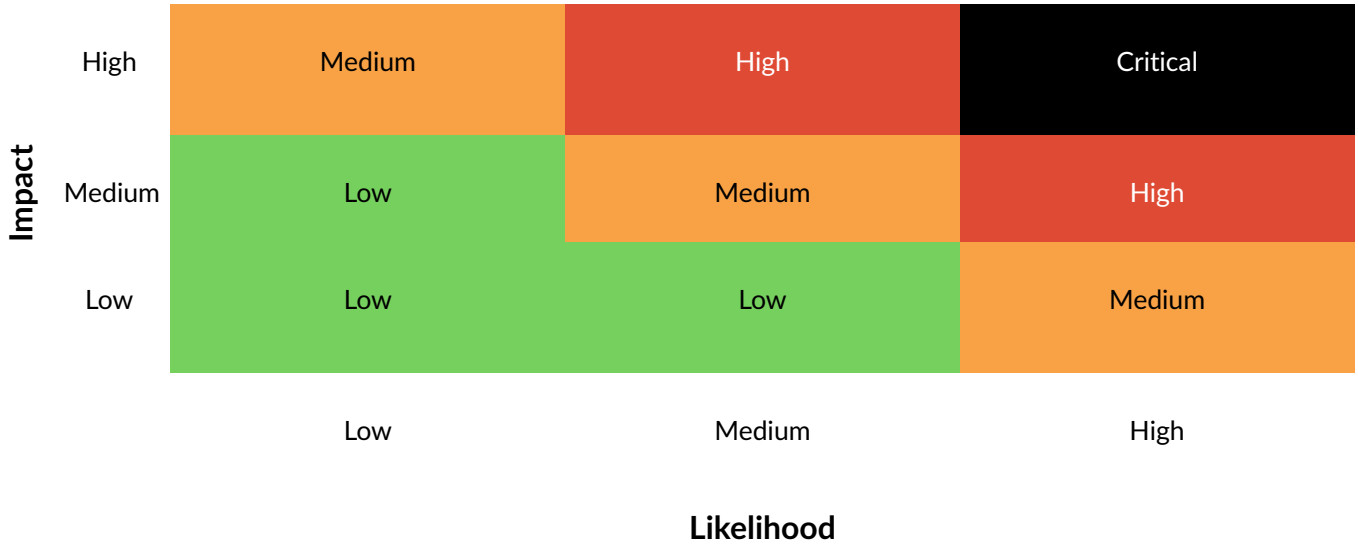

Table 1: Severity Matrix - How the severity of a vulnerability is given based on the *impact* and the *likelihood* of a vulnerability.